Refinement & Synthesis - Distributed Event Clock Automata

Ortiz Vega, James Jerson; Schobbens, Pierre-Yves; Legay, Axel

Publication date:
2011

Document Version
Early version, also known as pre-print

MoVES Newsletter

Work Package 5 - Refinement & Synthesis

University in Focus: Université Libre de Bruxelles

Moves Newsletter, No. 6, June 2011

Editorial

Dear Reader, The MoVES project on Modelling, Verification and Evolution of Software is sponsored by the Belgian government with its "Interuniversity Attraction Poles" (IAP) Programme which aims at providing support for teams of excellence in basic research that belong to Belgium's various (linguistic) Communities. These teams work as part of a network in order to increase their joint contribution to general scientific advances and, where applicable, to international scientific networks. Since 1987 the IAP programme has been implemented in five year phases. The ongoing IAP phase-VI (2007-2011) contains 44 networks implicating 324 research teams (250 Belgian teams and 74 European partners) and spans a wide variety of research fields in the life sciences, exact and applied sciences and the human sciences. The impact of the IAP on basic research is considerable. The programme represents some 500 researchers (full-time-equivalents) paid with IAP funds and over 2 000 publications each year within the programme as a whole. **At the time of this writing, we still do not know if the whole IAP programme will continue beyond 2011 or will simply be deleted.**

The Moves project combines the leading Belgian teams in software engineering and formal design and verification of software. This month, we focus on the work achieved so far inside package 5 on Refinement and Synthesis. It is also the opportunity to present related research that is conducted by the University of Brussels (ULB).

This newsletter contains contributions by Ortiz et al. on distributed event clock automata, Combéfis et al. on human machine interaction interfaces, Doyen et al. on games with imperfect information, Kalyon et al. on distributed controller synthesis, Geeraerts et al. on timed games, Shirmohammadi et al. on objectives and winning strategy in stochastic games, and Filiot et al. on Acacia, a tool for LTL synthesis.

Enjoy reading!

Thierry Massart and Jean-François Raskin

Upcoming Events & Recent Publications

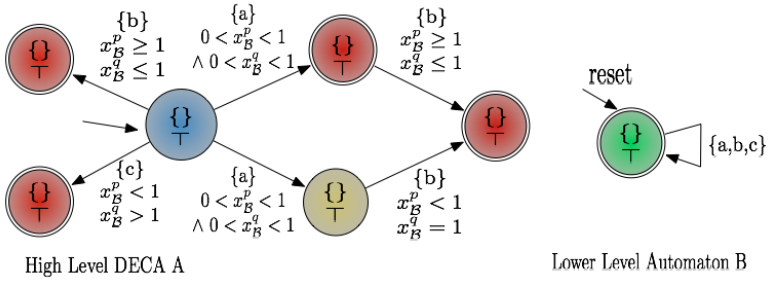
- *Sample of recent publications:*
  - Laurent Doyen and Jean-François Raskin. Games with Imperfect Information: Theory and Algorithms. Lectures in Game Theory for Computer Scientists. Cambridge University Press, pages 185-212, 2011.
  - Laurent Doyen, Thierry Massart, and Mahsa Shirmohammadi. Synchronizing Objectives for Markov Decision Processes. Proceedings of International Workshop on Interactions, Games and Protocols (iWIGP), Electronic Proceedings in Theoretical Computer Science 50, pages 61-75, 2011.
  - Barbara Di Giampaolo, Gilles Geeraerts, Jean-François Raskin, and Nathalie Sznajder. Safraless Procedures for Timed Specifications. In FORMATS'10, LNCS 6246, Springer, pages 2-22, 2010.
  - Emmanuel Filiot, Tristan Le Gall, Jean-François Raskin. Regret Minimization in Game Graphs. In MFCS'10, LNCS 6281, Springer, pages 342-354, 2010.
- *Upcoming Workshop*: 3rd Workshop on Games for Design, Verification and Synthesis Colocated with CONCUR'11 – Aachen (Germany), 10 September 2011 (`http://www.lsv.ens-cachan.fr/Events/gasics11/`)

James Ortiz, Axel Legay, Pierre-Yves Schobbens

Distributed Event Clock Automata

Real-Time Distributed Systems (RTDS) take an increasingly important role in our society, including in aircrafts and spacecrafts, satellite telecommunication networks or positioning systems. Distributed Systems consist of computer systems at different locations, that communicate through a network to achieve their function. Real-Time Systems have to obey strict requirements about the time of their actions. To ensure these, they rely on clocks. When systems are widely distributed, we cannot assume that their clocks are perfectly synchronized.

One of the most successful techniques for modeling real-time systems are Timed Automata (TA). A timed automaton is a finite automaton augmented with real-valued clocks. Constraints on these clocks are used to restrict the behaviors of the automaton. The model of TA assumes perfect clocks: all clocks have infinite precision and are perfectly synchronized. This causes TA to have an undecidable language inclusion problem.



To restore decidability, Alur et al. [1] proposed to restrict the behavior of clocks. The key idea is that the problematic clocks of TA are reset by non-deterministic transitions. In contrast, an event clock (EC) x_p is reset when a given atomic proposition p occurs. The event clock values are deterministic and thus Event Clock Automata ECA are determinizable, making language inclusion decidable and thus enabling refinement based development. Event clocks can also be introduced in temporal logic. An event clock constraint is naturally translated into a proposition $\lhd_I p$, that means "the last time that a p occurred was d time units ago, where d lies in I".

However, the expressiveness of ECA is rather weak. Furthermore, this logic violates the substitution principle: any proposition should be replaceable by a formula. In 1999, Raskin [2] introduced the notion of "recursive" event. In a recursive event model, the reset of a clock is decided by a lower-level automaton (or formula). This automaton cannot read the clock that it is resetting. Clock resets are thus still deterministic, but the concept of "event" is now much more expressive. $\rhd_I$ and $\lhd_I$ are modalities that can contain any subformulas, and can be nested. The temporal logic of recursive event clocks (EventClockTL) has the same expressiveness as Metric Interval Temporal Logic MITL in the interval semantics.

Based on these extensions, we removed the assumption of perfect clock synchronization, inspired by Bengtsson et al. [3] studied the worst case: where the clocks can advance totally independently if they are in different processes. In this direction, we have proposed Distributed Recursive Event Clock Automata (DECA). We have shown that DECA are determinizable, thus closed under complementation, and thus that their language inclusion problem is decidable (PSPACE-complete). We have defined Distributed Event Clock Temporal Logic (DECTL), which is also PSPACE-complete.

[1] Rajeev Alur, Limor Fix and Thomas A. Henzinger. A Determinizable Class of Timed Automata. In CAV'94, pages 1-13, 1994.

[2] Jean-François Raskin. Logics, Automata and Classical Theories for Deciding Real Time. PhD thesis, FUNDP University, Belgium, 1999.

[3] Johan Bengtsson, Bengt Jonsson, Johan Lilius and Wang Yi. Partial Order Reductions for Timed Systems. In CONCUR'98, pages 485-500, 1998.

Further Information: *http://www.info.fundp.ac.be/~acs/tvl/*

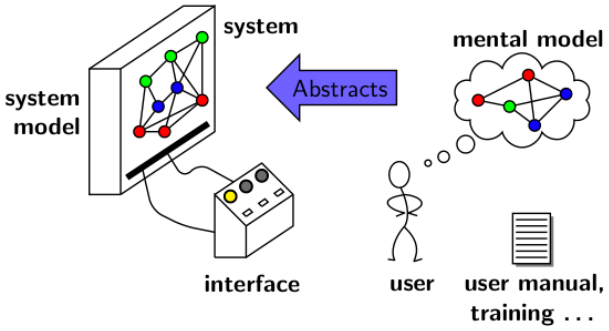
Contact: *jor@info.fundp.ac.be, alegay@irisa.fr, pys@info.fundp.ac.be*

Sébastien Combéfis, Charles Pecheur

HMI Analysis

In many safety-critical applications, such as flight control systems in civil aviation, uncertainty or confusion about the state of the system state in the mind of its human operator can lead to anomalous and even hazardous situations. These issues, rooted in system observability and abstraction concerns, can be addressed through appropriate formal modelling and analysis techniques. The LVL group in UCL is studying the use of formal techniques for the analysis of human-machine interactions (HMI). Our focus is on generating system abstractions for human operators, which we call "mental models". Such mental models, once expressed in rigorous, formal notations, can be used for analysis or for user training. They should ideally be *minimal* in order to concisely capture the behavior of the system from the point of view of an operator. They should also allow *full control*, which ensures that they contain enough information to allow proper control of the system.

We investigate the problem of automatically generating minimal full-control mental models, based on formal descriptions of the system's behaviour. In a first approach, we have proposed a variant of Paije and Tarjan's classical coarsest partition algorithm to obtain the mental model by minimizing the system model. More recently, we have been studying an alternative approach based on machine learning techniques. In this approach, (a 3-value variant of) the L^* learning algorithm is taught to learn the mental model for a given system. Moreover, if no adequate mental model can be generated due to insufficient observability in the system, resulting diagnostic information can be used to identify and resolve the issue. The well-known HMI issue of mode confusion can be analyzed through this approach.



Further Information: *http://lvl.info.ucl.be/*
Contact: *sebastien.combefis@uclouvain.be, charles.pecheur@uclouvain.be*

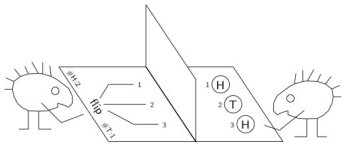
University in Focus – Université Libre de Bruxelles

Laurent Doyen and Jean-François Raskin

Games with Imperfect Information

Games played on graphs are natural models for reactive systems. The graph describes the possible interactions of the system with the environment, and the game is of infinite duration because reactive systems are usually not expected to terminate. In the simplest setting the players have full knowledge of both the game structure and the sequence of moves played by the adversary. The winning condition is defined by a set of plays (paths in the graph) that the first player aims at enforcing, and that the second player aims at avoiding. This is however unrealistic in the design of reactive systems because the components of a system have an internal state that is not visible to the other components. Such situations require to introduce games with imperfect information where the players have partial information about the play. We illustrate the games with imperfect information with the 3-coin game (illustrated by the drawing).

Three coins c_1, c_2, c_3 are arranged on a table, either head or tail up. Player 1 does not see the coins, but he is informed of the number of heads (H) and tails (T). The coins are manipulated by Player 2. The objective of Player 1 is to have all coins head up (HHH) while avoiding at all cost a configuration where all coins show tail (TTT). The game is played as follows. Initially, Player 2 chooses a configuration of the coins with two heads and one tails. Then, the following rounds are played: Player 1 can choose one coin in the set $\{c_1, c_2, c_3\}$ and ask Player 2 to toggle that coin. Player 2 must execute the choice of Player 1 and he may further decide to exchange the positions of the two other coins. The game stops whenever the three coins are all head up (Player 1 wins) or all tail up (Player 2 wins). Otherwise Player 2 announces the number of heads and tails, and the next round starts. This game can easily be modeled as a game graph whose states are configurations of the coins and the observation associated to a state is the number of coins that are H and T.



We have designed algorithms that given a game of imperfect information, decide if Player 1 has: (i) a deterministic observation-based strategy to win, or (ii) a randomized observation-based strategy to win. In our example, Player 1 has no deterministic observation-based strategy to win the 3-coin game, because Player 2 can always find a spoiling counter-strategy using his ability to exchange coins after Player 1's move. If we do not allow Player 2 to exchange the coins, then Player 1 has a deterministic observation-based winning strategy consisting in successively trying to toggle every coin. This strategy requires memory and it is easy to see that memory is necessary to win this game. On the other hand, if we allow Player 1 to take his decision using a source of randomization, then she can win the original 3-coin game with probability 1 (do you see why?). This shows that randomized strategies are in general more powerful than deterministic strategies.

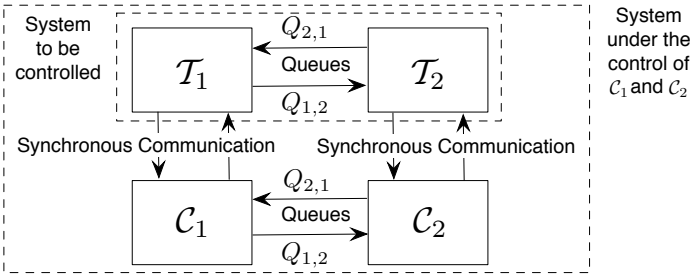
Details about this work can be found in: Laurent Doyen and Jean-François Raskin. Games with Imperfect Information: Theory and Algorithms. Lectures in Game Theory for Computer Scientists. Cambridge University Press, pages 185-212, 2011.

Further Information: *http://www.ulb.ac.be/di/verif/index-en.html*
Contact: *doyen@lsv.ens-cachan.fr,jraskin@ulb.ac.be*

Gabriel Kalyon, Tristan Le Gall, Hervé Marchand, Thierry Massart

Synthesis of Communicating Controllers for Distributed Systems

We consider the control of distributed systems composed of subsystems communicating asynchronously; the aim is to build local controllers that restrict the behavior of a distributed system in order to satisfy a global state avoidance property. We model our distributed systems as *communicating finite state machines* with reliable unbounded FIFO queues between subsystems. The local controllers can only observe the behavior of their proper local subsystem and do not see the queue contents. To refine their control policy, the controllers can use the FIFO queues to communicate by piggybacking extra information (some timestamps and their state estimates) to the messages sent by the subsystems. We provide an algorithm that computes, during the execution of the system, an estimate of the current global state of the distributed system for each local subsystem (and thus for each controller). When a message is received by a subsystem, its local controller uses this information to compute a better approximation of the global state estimate so as to obtain a more permissive control policy. We then define synthesis algorithms allowing to compute the local controllers. Our method relies on the computation of (co)reachable states. Since the reachability problem is undecidable in our model, we use abstract interpretation techniques to obtain regular overapproximations of the possible FIFO queue contents, and hence of the possible current global states. An implementation of our algorithms provides an empirical evaluation of our method.



Further Information: *http://www.ulb.ac.be/di/verif/*
Contact: *gkalyon@ulb.ac.be, tlegall@ulb.ac.be, herve.marchand@inria.fr, tmassart@ulb.ac.be*

Gilles Geeraerts and Jean-François Raskin

Timed Games

Embedded controls have often to meet real-time constraints. Timed automata were introduced by Rajeev Alur and David Dill [1] in early 90's as a formal model for real-time systems. Timed automata extend plain finite state automata with real-valued clocks that count time, that can be reset and whose value can be compared in guards against constants. To synthesize controllers, timed automata have been extended to timed game automata. In a timed game automaton, transitions are partitionned as being controllable or uncontrollable. As the actions taken by the environment are uncontrollable, those actions should be considered as *adversarial*. Indeed, a controller should be correct no matter how the environment in which it operates behaves.

Let us consider the example of the Chinese juggler control problem to illustrate the notion of timed game. A Chinese juggler has to spin plates that are on sticks to prevent them from falling, see the drawing for an illustration. Assume, for our example, that the juggler wants to handle two plates, called Plate 1 and Plate 2. Plates crash after a while if they are not spun. Initially, each plate is spinning on its stick and the spin is fast enough so that they will stay stable for at least 5 seconds. The juggler has to maintain them stable for as long as possible (forever if possible). For that, the juggler can spin each plate but he can spin only one of the plates at a time. When he decides to spin Plate $i \in \{1, 2\}$, he should do it for at least 1 time unit. If he decides to do it for t time units then Plate i stays stable for 3 time units if $1 \leq t \leq 2$, and for 5 time units if $t > 2$.

Now, assume that there is also a mosquito in the room. When the mosquito touches one of the two plates, it reduces the spinning of the plate, and as a result its remaining stability time is decreased by 1 time unit. When the mosquito touches the plate, it gets afraid and this guarantees that it will not touch any plate again before D time units have elapsed (after that amount of time the mosquito has forgotten and he is not afraid any more). Can you answer the following question: "Given a value for D, does the Chinese juggler have a way to spin the plates so that none of the two plates ever falls down no matter what the behavior of the mosquito is ?"

The Chinese juggler control problem can easily be modeled using timed game automata, and the tool UPPAAL-TIGA (`http://www.cs.aau.dk/~adavid/tiga/`) can compute for you a winning strategy for the juggler against any behavior of the plates and the mosquito.

Those techniques have been applied successfully to the automatic synthesis of optimal timed controllers in industrial applications. The interested reader will find more information about the possible applications of automatic synthesis in the following paper: Franck Cassez, Jan J. Jessen, Kim Larsen, Jean-François Raskin and Pierre-Alain Reynier. Automatic Synthesis of Robust and Optimal Controllers - An Industrial Case Study. In HSCC'09, LNCS 5469, Springer, pages 90-104, 2009.

[1] Rajeev Alur and David L. Dill. A theory of timed automata. Theoretical Computer Science, 126(2):183–235, 1994.

Further Information: *http://www.ulb.ac.be/di/verif/index-en.html*
Contact: *gilles.geeraerts@ulb.ac.be,jraskin@ulb.ac.be*

Mahsa Shirmohammadi, Laurent Doyen, Thierry Massart

Quantitative Verification for Probabilistic Systems

Qualitative verification refers to the traditional boolean one where a (behavior of a) system is evaluated as either (good or bad) correct or incorrect. However, the reality is often richer, for instance among two correct systems, one may be more desirable than the other; or for an incorrect system, it may be useful for a designer to know how far it is from a correct solution. Moreover, the qualitative view is not robust since a slight change in the behavior of a correct system may turn it into an incorrect one. Therefore, we advocate a richer notion of verification, namely quantitative verification. We investigate new models of computation for the quantitative verification for reactive systems. Specifically, we consider these questions in the context of stochastic systems modeled by stochastic games. While Markov decision processes (MDPs) are a well-established syntax for describing such reactive systems, we study new semantics which are suitable for expressing new quantitative objectives.

In previous works, we have defined a quantitative objective called synchronizing objectives (for MDPs) which is based on sequences of probability distributions. Intuitively, a synchronizing objective requires that eventually, at every step there is a state which concentrates almost all the probability mass. In particular, it implies that the probabilistic system behaves in the long run like a deterministic system. Defining objectives as a sequence of probability distributions over states rather than a distribution over sequences of states is a change of standpoint in the traditional approach to probabilistic verification. Since a sequence of probability distributions is in essence a quantitative object (through the real numbers), this is a natural and promising framework to understand how to specify and characterize quantitative counterparts of classical boolean notions.

We will generalize these new types of objectives for stochastic games with more players where studying the standard notions such as determinacy and equilibrium are also of interest. Determinacy refers to the fact that a winning strategy always exists for one of the players in games with strictly conflicting objectives. For not necessarily conflicting objectives, several solution concepts usually called equilibrium have been considered, such as the Nash equilibrium. As another direction, we will explore the relation of this new semantics with classical notions of quantitative objectives such as limit-average and discounted.

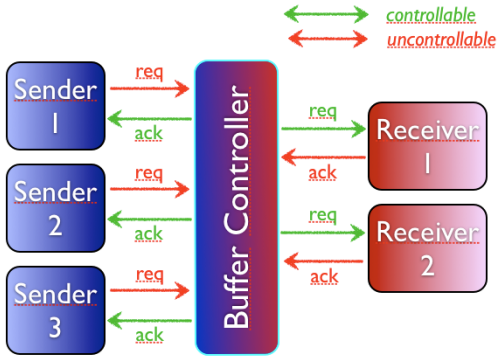
Further Information: *http://www.ulb.ac.be/di/verif/*
Contact: *mahsa.shirmohammadi@gmail.com, doyen@lsv.ens-cachan.fr, tmassart@ulb.ac.be*

Emmanuel Filiot, Naiyong Jin, Jean-François Raskin

Acacia: a Tool for LTL Synthesis

A *reactive module* is a system that interacts with some environment. At each time point, it receives some *uncontrollable* input signals from the environment and produces (controllable) output signals.

The figure shows some reactive module: a buffer controller. The senders may request the module to transmit some data to a receiver by sending the signal *req*, and the module may ackwnoledge the request by sending the signal *ack*. The module may also request a receiver to receive a data by sending the signal *req* and the receiver may grant the request by sending *ack*. The buffer controller cannot know when a request will be sent by a sender neither when a receiver will acknowledge a request, therefore those signals are uncontrollable. The executions of such a system are represented by the infinite sequences of signals that are sent.



Given a specification of a set of *good* executions, the *synthesis problem* asks to automatically generate a reactive module whose executions satisfy the specification. The *linear-time temporal logic (LTL)* is well-suited to describe temporal properties of reactive modules, and has been first studied in the context of synthesis by Pnueli and Rosner in 1989 [1]. While the LTL synthesis problem is 2-ExpTime-Complete, there has been a renewed interest in it and recent works have shown its practical feasibility.

We have developed the tool *Acacia* for the synthesis of reactive modules. It takes as input an LTL specification and outputs, if it exists, a finite-state Moore machine that realizes the specification. The underlying algorithm is based on a reduction to games played on graphs. Indeed, as the environment behavior is uncontrollable, it is natural to see the LTL synthesis problem as a game where the environment (the adversary) plays against the system (the protagonist). The system wins if it has a strategy such that whatever the environment does, the system executions which are consistent with the strategy satisfy the given LTL specification. Therefore a winning strategy is nothing else than a reactive module that fulfills the specification. Moreover, we have shown that *safety* games, i.e. games where the protagonist wins if it has a strategy to avoid a given subset of game positions, are sufficient to reason about LTL specifications. Moreover, our safety games are well-suited to support compositional synthesis algorithms and have a nice structure which allows one to compactly represent sets of winning positions. Our tool Acacia can be fed with several subspecifications and tries to synthesize a reactive module for the parallel composition of all those specifications. This allows one to synthesize specifications that are *several pages* long. On the other hand, when the specification is not realizable, Acacia can output a counter-strategy for the environment that can help the user in the task of correcting the specification.

[1] Amir Pnueli and Roni Rosner. On the Synthesis of a Reactive Module. In POPL'89, pages 179-190, 1989.

Further Information: *http://www.antichains.be/acacia/*
Contact: *efiliot@ulb.ac.be, naiyjin@ulb.ac.be, jraskin@ulb.ac.be*

Partners


Universiteit
Antwerpen


FUNDP
NAMUR


UMH


MoVES
Modelling, Verification and Evolution of Software


Vrije
Universiteit
Brussel


KATHOLIEKE UNIVERSITEIT
LEUVEN


ULg
UNIVERSITÉ de Liège


ULB


INRIA
centre de recherche
LILLE - NORD EUROPE


Imperial College
London


TU Delft
Delft
University of
Technology